QCard DLL Lesson 12

Lesson 11 Lesson 13

- Alyce

QCard DLL Lesson 12 | Dragging Cards Automatically | Dude, where's my card? | Is the move legal? | ReturnDrag | DEMO

See Lesson 1 for QCard DLL and WAV files needed for the demo code.

Dragging Cards Automatically

QCard DLL has a function that automatically drags a card from its current position to the xy position specified. It is called ReturnDrag. It might be used to return acard to its original location after a user attempts to drop it in an illegal spot. It can also be used to move a card so that it lines up properly in a row or column or on a pile of other cards. It might also be used for special effects.

Dude, where's my card?

If we want to return a card to its original location, we need to know the values for x and y. We can do this with GetCardX and GetCardY in the DLL.

```
calldll #qc, "GetCardX",_
    nC as long,_    'index of card
    CardX as long   'x location of upper corner

calldll #qc, "GetCardY",_
    nC as long,_    'index of card
    CardY as long   'y location of upper corner
```

InitDrag returns the index of the card under the mouse, so we can use that value to get the X and Y location of the card.

```
cardIndex=InitDrag(hBox,MouseX,MouseY)
```

We'll use the return from InitCard to see if the card is blocked. If it is blocked, we'll call AbortDrag.

```
    isBlocked=GetCardBlocked(cardIndex)    'see if card is blocked
    if isBlocked<>0 then
        'if card is blocked, inform user and abort drag operation
        call AbortDrag
        #1.g
"place 10 420;\BLOCKED Card Index is ";cardIndex;space$(60)
        wait
    end if
```

If the card is not blocked we'll allow the program to continue. We'll keep track of the index of the dragged card in a variable called dragCard and we'll get the X and Y location in case we need to return the card later.

```
    'document card index and location for return drag
    dragCard=cardIndex
    cardX=GetCardX(dragCard)
    cardY=GetCardY(dragCard)
```

Is the move legal?

For this demo we've eliminated the shuffle to simplify things. We've also drawn a box in the corner as a target for the user to drag a card. Cards must land in the box or the move is illegal.

We allow drag. When the user releases the right mouse button we check with GetCardX and GetCardY to see if the card is completely within the borders of the box.

```
    #1.g "when rightButtonUp"
    cardIndex=EndDrag(hBox,MouseX,MouseY)
    cX=GetCardX(dragCard)  'get x location of card
    cY=GetCardY(dragCard)  'get y location of card
```

Here is the routine that checks the location and uses the ReturnDrag function to return the card if it is not in the box. If it is within the box, we use ReturnDrag to line it up with previous cards in the pile.

```
    if (cX<450) or (cY<250) then
        'return card if it is not in box
        call ReturnDrag hBox, dragCard,cardX,cardY
```

```
        #1.g "place 10 420;\Illegal move.";space$(60)
    else
        'put it on pile in box
        call ReturnDrag hBox, dragCard, 470, 260
        #1.g "place 10 420;\Card placed in box. Thanks. ";space$(50)
    end if
```

ReturnDrag

Here is the ReturnDrag function which causes the automatic dragging operation. It requires the handle of the graphicbox, the index of the card to drag, and the xy target location.

```
    calldll #qc, "ReturnDrag",_ 'automatic dragging
        hndle as ulong,_    'handle of graphicbox
        nC as long,_        'card to drag
        nx as long,_        'x location to drag to
        ny as long,_        'y location to drag to
        re as void          'no return
```

DEMO

See Lesson 1 for QCard DLL and WAV files needed for the demo code.

```
'An open project card game, begun by Alyce Watson, May 27, 2003.
'Uses Qcard32.dll, a freeware library of playing card images.
'DLL by Stephen Murphy.  Qcard32.DLL website:
'http://www.telusplanet.net/public/stevem/

dim card(24)            'array to hold cards
gosub [fillCardArray]   'fill array with card values

[varSetup]
i=0         'i will be our counter var in for/next loops
design=1    'default design is circles
offset=20   'offset for card vertical spacing for dragging purposes

nomainwin
    WindowWidth=640:WindowHeight=480
    UpperLeftX=1:UpperLeftY=1

    menu #1, "&File", "&New",[new],"E&xit", [quit]
```

```
    graphicbox #1.g, 0, 0, 640, 440
    open "Dragging Cards" for window_nf as #1
    #1 "trapclose [quit]"

    'get graphicbox handle
    hBox=hwnd(#1.g)

    'open the dll
    open "qcard32.dll" for dll as #qc
    'initialize the deck
    Call InitializeDeck hBox

[new]
    Call SetDefaultValues

    'draw a nice background
    #1.g "down; fill 10 190 225"
    #1.g "backcolor 10 190 225"
    'draw a box in lower right corner
    #1.g "place 450 250;box 700 500"
    #1.g "place 450 230;\Drag Cards Here."
    'temp message for this demo only
    #1.g "place 10 420"
    #1.g "\Right-
click on a card, then move mouse to drag and drop it."

    offset=20
    call SetOffSet offset    'set offset to 20 pixels - default is 16

    'set xy location to start deal
    x=10:y=2:row=1
    for i = 1 to 24
        Call DealCard hBox,card(i),x,y
        'When creating a row or pile of cards,
        'only the topmost card should have an
        'IsBlocked value of FALSE.
        if row<4 then call AdjustCardBlocked card(i),1

        x=x+100
        if x>510 then   'move to next row
            x=10
            y=y+offset
            row=row+1
        end if
        playwave "card.wav",sync
```

```
        'pause 100 milliseconds between cards
         call Pause 100
         scan
    next
    #1.g "when rightButtonDown [initDrag]"
wait

[initDrag]
    cardIndex=InitDrag(hBox,MouseX,MouseY)
    isBlocked=GetCardBlocked(cardIndex)    'see if card is blocked
    if isBlocked<>0 then
         'if card is blocked, inform user and abort drag operation
         call AbortDrag
         #1.g
"place 10 420;\BLOCKED Card Index is ";cardIndex;space$(60)
         wait
    end if
    #1.g "place 10 420;\Drag Card Index is ";cardIndex;space$(60)
    #1.g "when rightButtonMove [doDrag]"
    'document card index and location for return drag
    dragCard=cardIndex
    cardX=GetCardX(dragCard)
    cardY=GetCardY(dragCard)
    wait

[doDrag]
    call DoDrag hBox, MouseX, MouseY
    #1.g "when rightButtonUp [endDrag]"
    wait

[endDrag]
    #1.g "when rightButtonUp"
    cardIndex=EndDrag(hBox,MouseX,MouseY)
    cX=GetCardX(dragCard)  'get x location of card
    cY=GetCardY(dragCard)  'get y location of card

    if (cX<450) or (cY<250) then
         'return card if it is not in box
         call ReturnDrag hBox, dragCard,cardX,cardY
         #1.g "place 10 420;\Illegal move.";space$(60)
    else
         'put it on pile in box
         call ReturnDrag hBox, dragCard, 470, 260
         #1.g "place 10 420;\Card placed in box. Thanks. ";space$(50)
    end if
    wait
```

```
[fillCardArray]
    'fill card array
    'cards 1 to 52 are in the first deck
    'cards 53 to 104 are in the second deck
    'use cards Jack through King in each suit, first deck
    card(1)=11  'jack of clubs
    card(2)=12  'queen
    card(3)=13  'king
    card(4)=24  'jack of diamonds
    card(5)=25  'queen
    card(6)=26  'king
    card(7)=37  'jack of hearts
    card(8)=38  'queen
    card(9)=39  'king
    card(10)=50  'jack of spades
    card(11)=51  'queen
    card(12)=52  'king

    'now use second deck, to fill second half of array
    for i = 1 to 12
        card(i+12)=card(i)+52
    next
    RETURN

[quit] close #qc:close #1:end


''''''''''''''''''''
'subs and functions:
Sub Pause ms
    'pause ms number of milliseconds
    calldll #kernel32,"Sleep",_
    ms as long, re as void
    End Sub

Function GetCardX(nC)
    calldll #qc, "GetCardX",_
        nC as long,_    'index of card
        GetCardX as long 'x location of upper corner
    end function

Function GetCardY(nC)
    calldll #qc, "GetCardY",_
        nC as long,_    'index of card
        GetCardY as long 'y location of upper corner
```

```
    end function

Sub SetOffSet offset
    calldll #qc, "SetOffSet",offset as long,_
        re as void
    end sub

Sub AdjustCardBlocked nC, bValue
    calldll #qc, "AdjustCardBlocked",_
        nC as long, bValue as long, re as void
    end sub

Function GetCardBlocked(nC)
    calldll #qc, "GetCardBlocked",nC as long,_
        GetCardBlocked as long
    end function

Sub ReturnDrag hndle,nC,nx,ny
    calldll #qc, "ReturnDrag",_ 'automatic dragging
        hndle as ulong,_    'handle of graphicbox
        nC as long,_        'card to drag
        nx as long,_        'x location to drag to
        ny as long,_        'y location to drag to
        re as void          'no return
    end sub

Function InitDrag(hndle, x, y)
    calldll #qc, "InitDrag",_
        hndle as ulong, x as long, y as long,_
        InitDrag as long
    end function

Sub AbortDrag
    calldll #qc, "AbortDrag",re as void
    end sub

Sub DoDrag hndle,x,y
    calldll #qc, "DoDrag",hndle as ulong,_
    x as long, y as long, r as void
    end sub

Function EndDrag(hndle,x,y)
    calldll #qc, "EndDrag",hndle as ulong,_
    x as long, y as long, EndDrag as long
    end function
```

```
Sub InitializeDeck hndle
    calldll #qc, "InitializeDeck",_
    hndle as ulong,r as long
    End Sub

Sub DealCard hndle,nC,x,y
    'places card on window whose handle is hndle at x,y
    'nC is number of card - 1-52 in first deck and
    '53-104 in second deck, if used
    calldll #qc, "DealCard",hndle as ulong,nC as long,_
    x as long,y as long,r as void
    End Sub

Sub SetDefaultValues
    'reset all card properties back to their default values.
    calldll #qc, "SetDefaultValues",r as void
    End Sub
```

QCard DLL Lesson 12 | Dragging Cards Automatically | Dude, where's my card? | Is the move legal? | ReturnDrag | DEMO

Lesson 11 Lesson 13